New, Flashier, Terminal (defaults) hacks
****

I was playing around with the defaults system of Mac OS X today and found some pretty interesting stuff...This stuff was tested only on Mac OS X build 4k33, and may or may not work on earlier or later builds. Also, it is possible that these hidden features are already known about, in which case I apologize. I know you're anxious to hear about it, so we'll get started right away! (For more information on procedure for finding this breaking news out, and some other tech-discussion see the end of this message).

The Dock
****
Probably the coolest hidden feature I found was in the Dock. I am sure you and your friends have watched and been impressed by the "genie" effect when minimizing windows. In fact, this effect is handled by the Dock (see post-lab discussion). The cool thing is, that some snooping around reveals 2 new effects! I will write the lines needed to activate the alternative Minimization effects so you can try them out yourself in the comfort of your own home.

```
defaults write com.apple.Dock mineffect genie
```
-This is the standard effect. Nothing too interesting with this one, but its nice to have, should you not like any of the other 2.

```
defaults write com.apple.Dock mineffect scale
```
-Now we're talking! This one is really cool...I wont spoil the fun and tell you how it looks, but the name should give you a hint

```
defaults write com.apple.Dock mineffect suck
```
-Again, try it out yourself.

I also found another dock defaults switch, the showshadow option. I am not entirely sure about its workings, but I'll give you all the information I've found out and get back to you if I find something else.

```
defaults write com.apple.Dock showshadow
```
-Shows the shadow. Duh! This applies a very weak shadow to the dock window, which actually is quite cool. As soon as the key exists, the effect takes place, no matter what its value is. This implies that it is part of a more complex structure. I've got indications of that the full format should follow this format (w/ default values):
```
{Style = Shadow; Height = 2; Radius = 2; Azimuth = 90; Ka = 0.40; Elevation ? 35; Saturate = 1.0; }
```
Though, I have not fully grasped how to use this, so if anyone knows how to deal with those options, post away!
To remove the shadow, use the following to remove the key from the com.apple.Dock entry.
```
defaults delete com.apple.Dock showshadow
```

Another thing that we can note, is that even though the code to do it appears to be in the executable, the previous "orientation" hack does not appear to work under 4k33, as has been previously noted by several sources.

After you have applied any of those "hacks", you need to restart the Dock...I usually do this using the unix kill command, but if you're uncomfortable doing that, you can just as well log out and back in again, or use the Process Manager application included in the Utilities folder. This same note goes for all the hacks in this file, but for the particular application in question.

The Finder (the Desktop)
****
I uncovered some hidden Finder options too...Those may not be all that interesting, but they certainly have some practical value. Enjoy!

defaults write com.apple.Finder AppleShowAllFiles 1
When this has a value of 1, hidden files are displayed by the finder (that means, files/folders beginning with a period, as well as files/folders that have their Mac OS hidden flag set). This could be very useful for the adventurous person, who would like to explore the UNIX file tree, without having to resort to the potentially intimidating Terminal. To reset the feature apply:
defaults write com.apple.Finder AppleShowAllFiles 0
or
defaults delete com.apple.Finder AppleShowAllFiles

defaults write com.apple.Finder Finder.HasTrash 1
Although this one was known before, I had not noted that it changed name to Finder.HasTrash instead of Desktop.HasTrash. They're the same anyway

defaults write com.apple.Finder Finder.HasDarkBackground 1* (?)
This switch exists, although I am clueless when it comes to its effect, or its usage. Help us all by sharing your knowledge!

defaults write com.apple.Finder Finder.HasLocalVolumes 0
Doesnt appear to have any effect. This used to work, didn't it...Well now with the drives on the desktop as default it doesn't appear to...That's a pity...

This was all I had to share with you now. I will probably be back later today or some time in this week with more features. I encourage you to do the same, using the instructions in "Post-Lab Discussion". OS X is teaming up to a really cool, and actually quite customisable, system! :-)

Post-Lab Discussion
****
Here is a brief discussion that will help you follow my steps and validate my results and possibly find more hidden defaults keys.

First a short description of the 'defaults' command. This is part of the 'man' description of the utility:
"defaults allows users to read, write, and delete Mac OS X user defaults from a command-line shell. Mac OS X applications and other programs use the defaults system to record user preferences and other information that must be maintained when the applications aren't running (such as default font for new documents, or the position of an Inspector panel). Much of this information is accessible through an application's Preferences panel (or the equivalent), but some of it isn't, such as the position of the Inspector panel. You can access this information with defaults.

User defaults belong to domains, which typically correspond to individual applications. Each domain has a dictionary of keys and values representing its defaults; for example, "Default Font" = "Helvetica". Keys are always strings, but values can be complex data structures comprising arrays, dictionaries, strings, and binary data. These data structures are stored as property lists; see PropertyList(5) for more information."

This outlines the function of the defaults command quite well. It is a system that can be used by applications, to save preferences in XML format, but also to provide additional features not accessible through the Preferences Panel or other GUI.

Feel free to explore 'man defaults' for more information on its usage if you are not aware of it already.

Procedure
Finding these hidden features, I used the following few steps:
* Find a suitable domain (= Application to explore)
* Locate its executable bundle and extract the proper executable file from it (the file

containing the actual code, as opposed to resources)
* Use the UNIX 'strings' utility to locate potentially interesting strings in the executable (this finds all strings, that is streams of characters 4 chars or longer, and is useful because the preference values the application searches for have to be hard coded into the executable)
* Read through the strings output and find any suspicious words (i.e. files that may be keys). One way of doing this, is to find a known key (such as titlesize in com.apple.Dock) and carefully read any strings appearing before or after that entry.
* Find appropriate values for the key. This could be difficult, but you can always try boolean values (1/0 or true/false) or numbers
* Try it out!
* Restart the application to see the effect

```
> cd /system/library/coreservices/dock.app/contents/macos
> ls
>>>>> Dock
> strings - Dock
>>>>> Lots of output...Here is a short extract:
>>>>> com.apple.dock
>>>>> version
>>>>> persistent-apps
>>>>> persistent-others
>>>>> magnification
>>>>> largesize
>>>>> tilesize
>>>>> autohide
>>>>> showhidden
>>>>> showforeground
>>>>> launchanim
>>>>> orientation
>>>>> bottom
>>>>> left
>>>>> right
>>>>> pinning
>>>>> middle
>>>>> start
>>>>> mineffect
>>>>> genie
>>>>> scale
>>>>> suck
>>>>> QuitFinder
>>>>> showshadow
>>>>> { Style = Shadow; Height = 2; Radius = 2; Azimuth = 90; Ka = 0.40; Elevation
= 35; Saturate = 1; }
> defaults write com.apple.Dock mineffect scale
> top
>>>>> <output>
> kill 1032
```

Done!

The fact that the genie effect is handled by the Dock poses a great difficulty for people trying to create alternative docks. I am sure most people appreciate the genie effect, but as you know the dock is not as liked. I guess this is up to the creative programmers to handle...